Design Principles for DIALS

DIALS Development Team

DIALS-3 Workshop

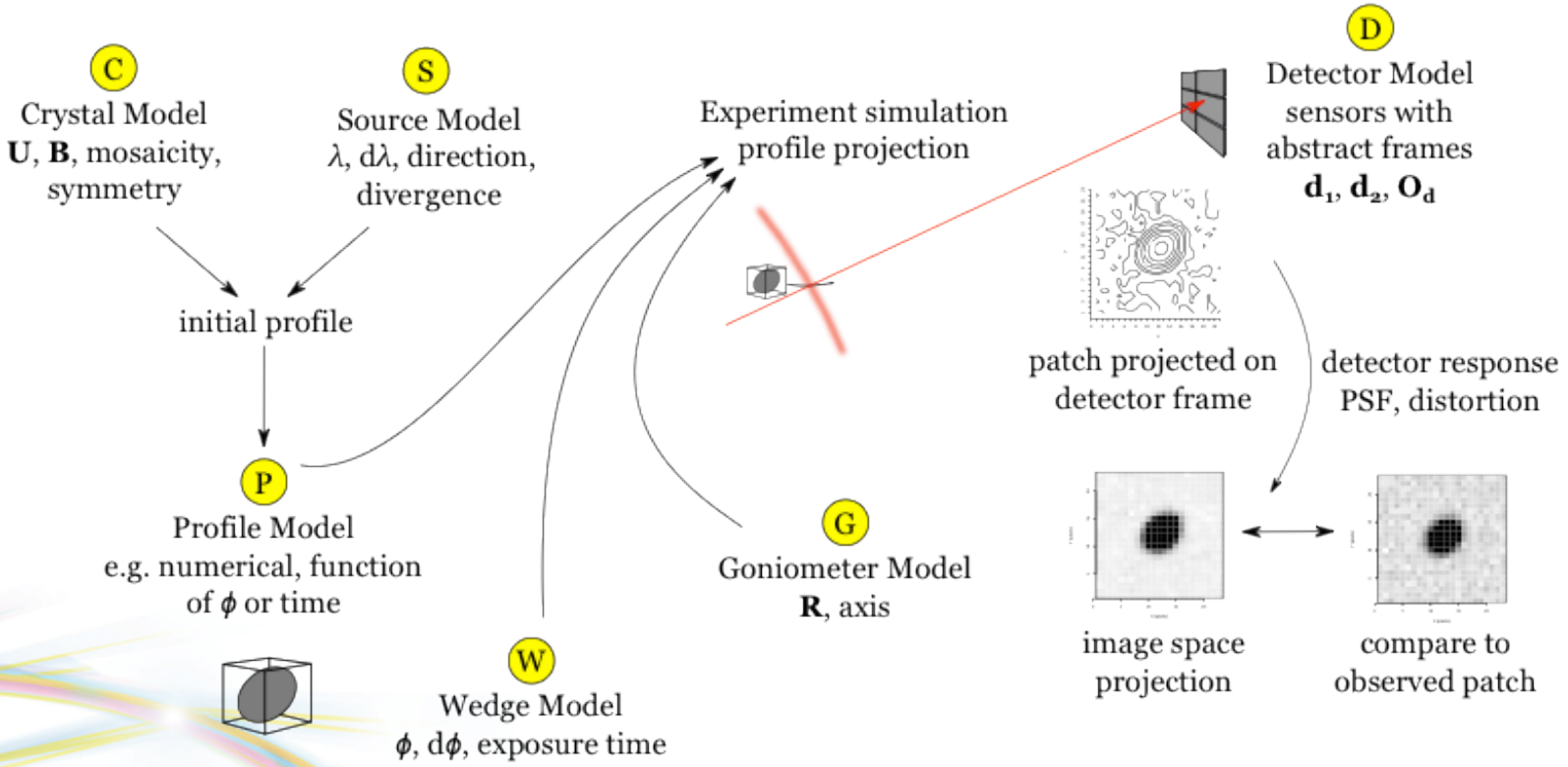
Preface

- Describing only developments on integration engine
- Work funded by EU through Biostruct-X work package 6, Diamond Light Source and CCP4
- Builds on massive amount of existing literature
- Working with LMB, Gleb Bourenkov

Overview

- "Top level" view
- Design principles
  - Global models, global refinement
  - Use of interfaces
  - Run-time extensibility
  - Flexibility of use
- Benefits of design
- Plans

Top-level view (one of our goals)



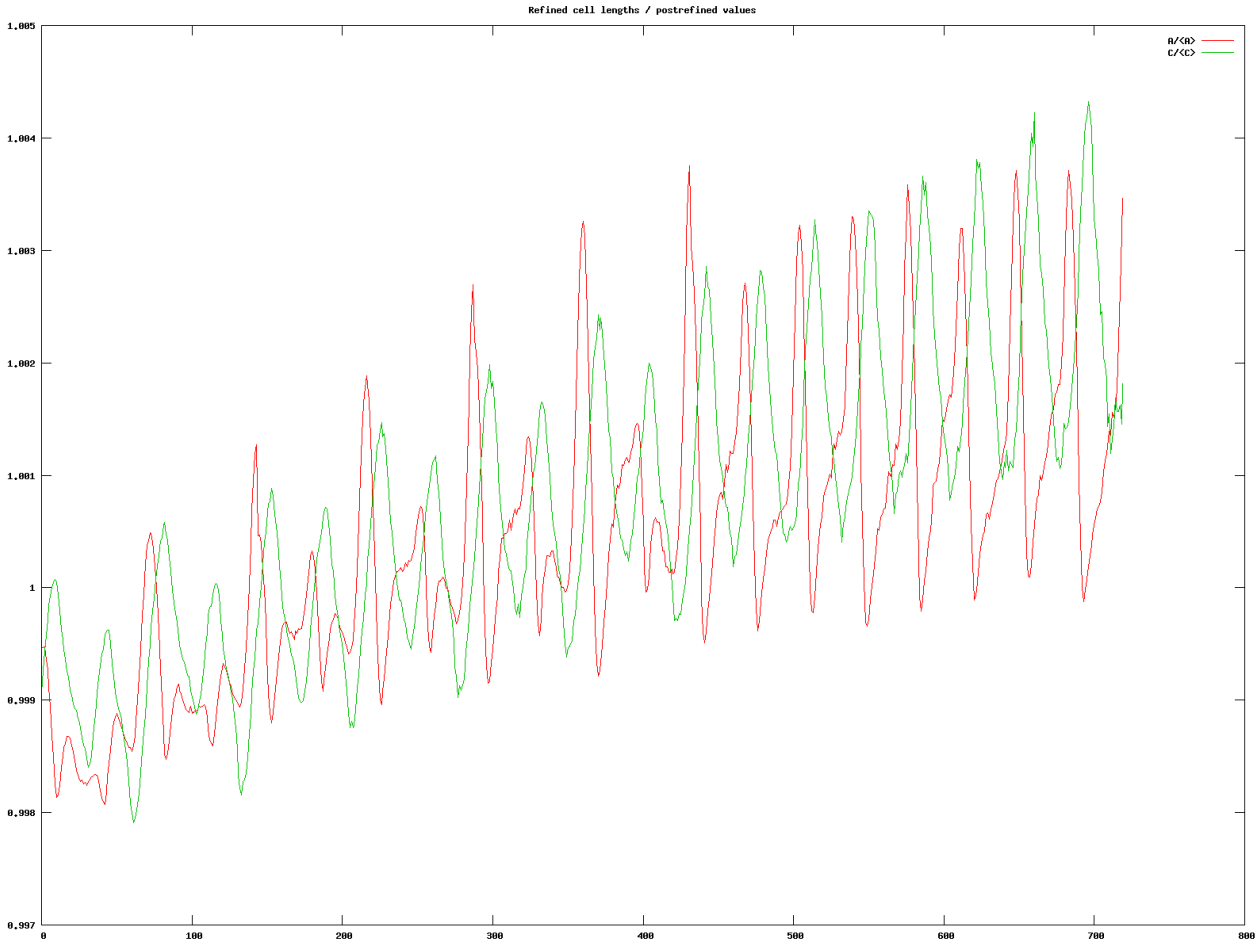
CCP4

diamond

Global model, global refinement

- Mosflm, XDS have “local” model, continuous refinement
  - Model to integrate frame i different to model for frame j
  - Harder to reliably parallelize to get identical results to serial code
- Have global (time dependent) model for all parameters => can run analysis in parallel

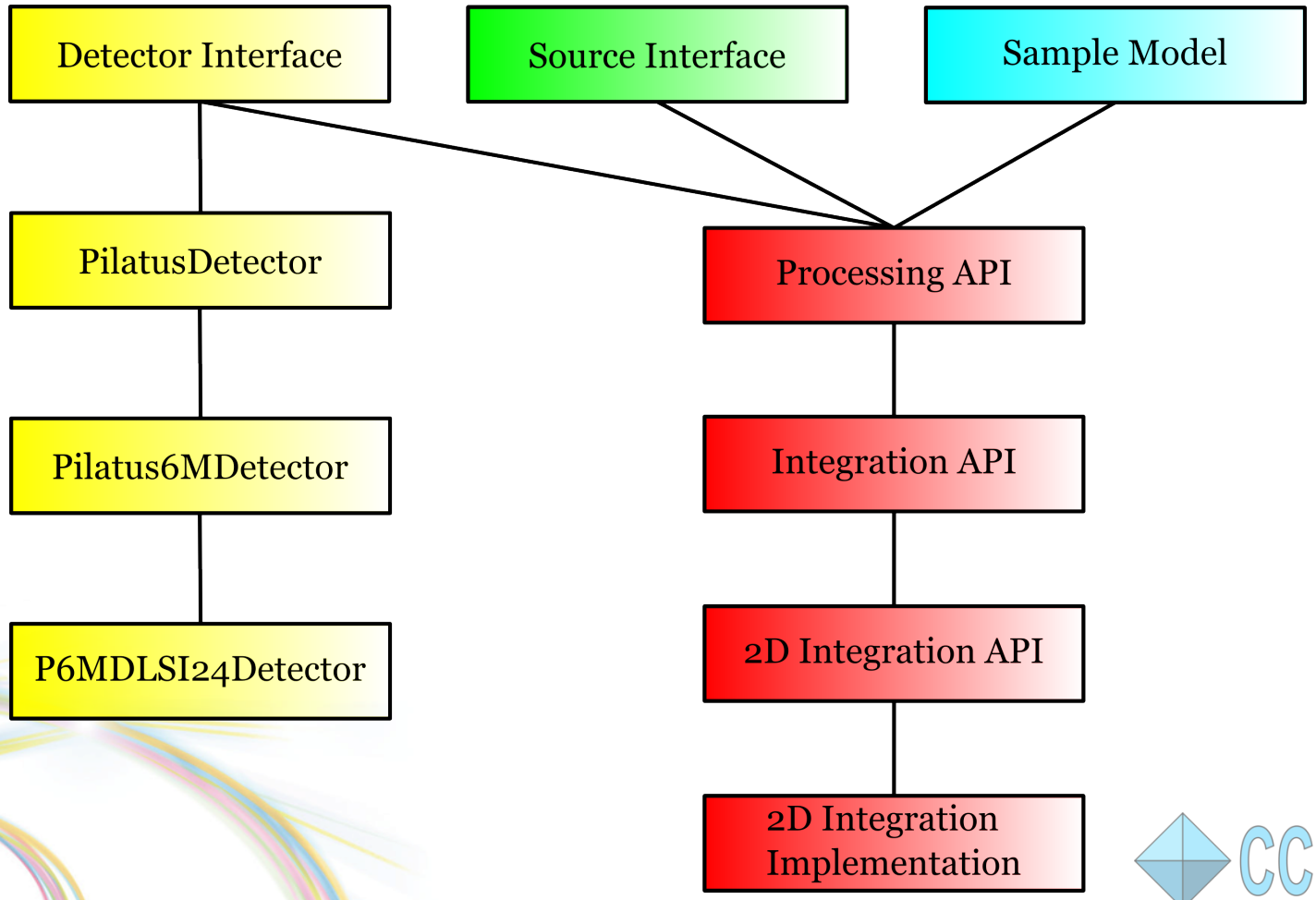
XDS processing 7200 degree set



CCP4

diamond

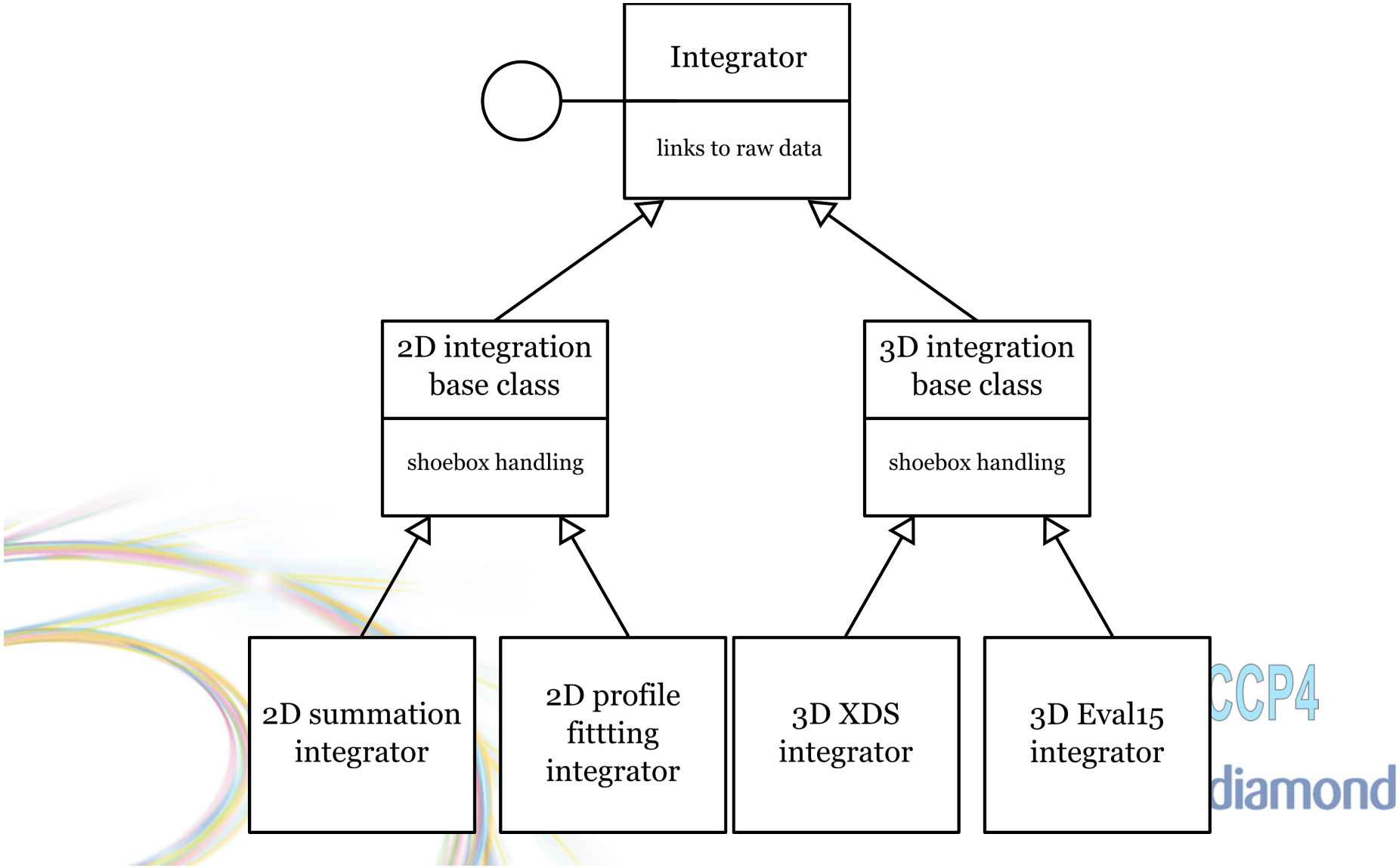
Design principles (interfaces)



CCP4

diamond

Design principles (interfaces)



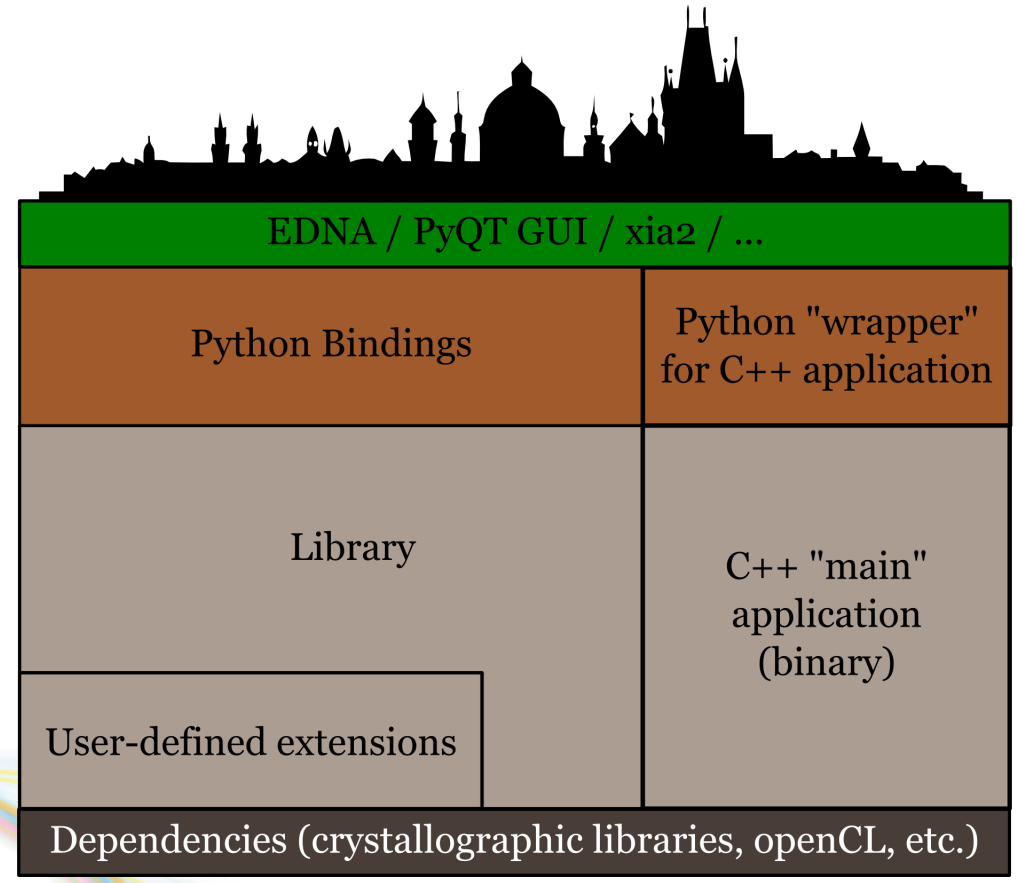
Interface designs

- Interfaces designed now will be wrong, though experience from xia2 will help
- Learning by doing, then design interfaces, then modify implementations to fit
- Take a light-weight approach to modeling – no UML, just describe in source code

CCP4

diamond

Design principles (extensibility)



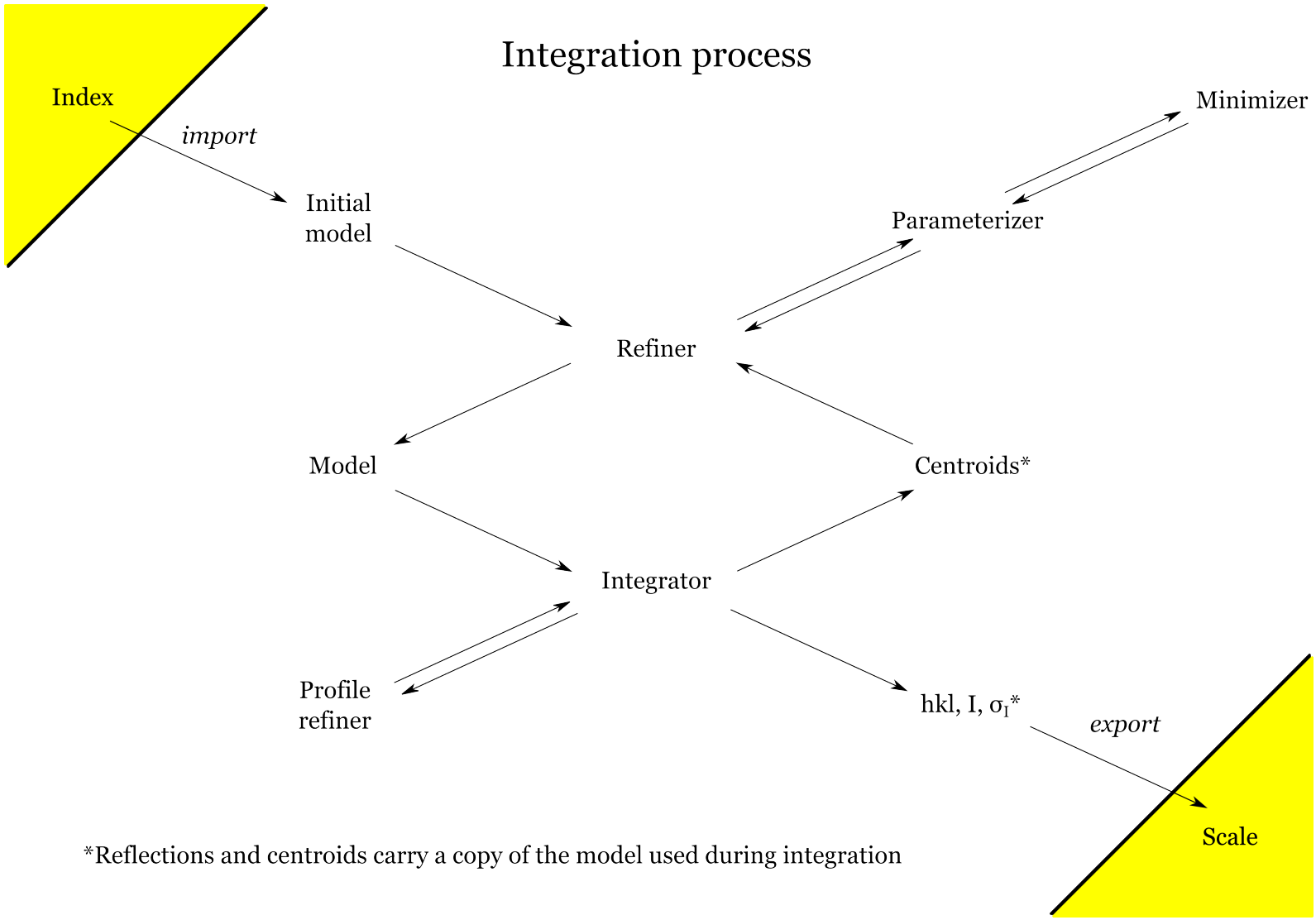
CCP4

diamond

Design principles (extensibility)

- This can actually work: use smart inheritance, registry, factory
- Example from dxtbx:
  - Define metaclass for interface
  - Write python code inheriting from interface
  - Registry searches for and sources python files
  - Metaclass auto-registers code, factory to recover
- Dependency on clean interfaces

Design principles (flexibility)



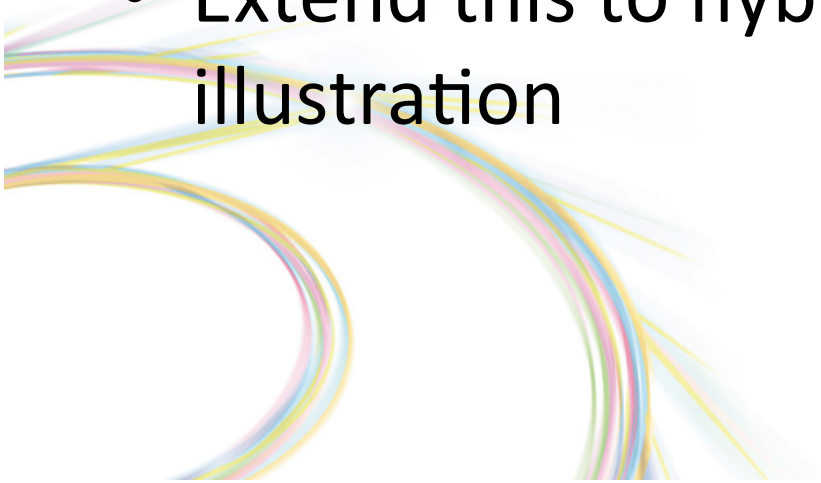
*Reflections and centroids carry a copy of the model used during integration

Benefits of design

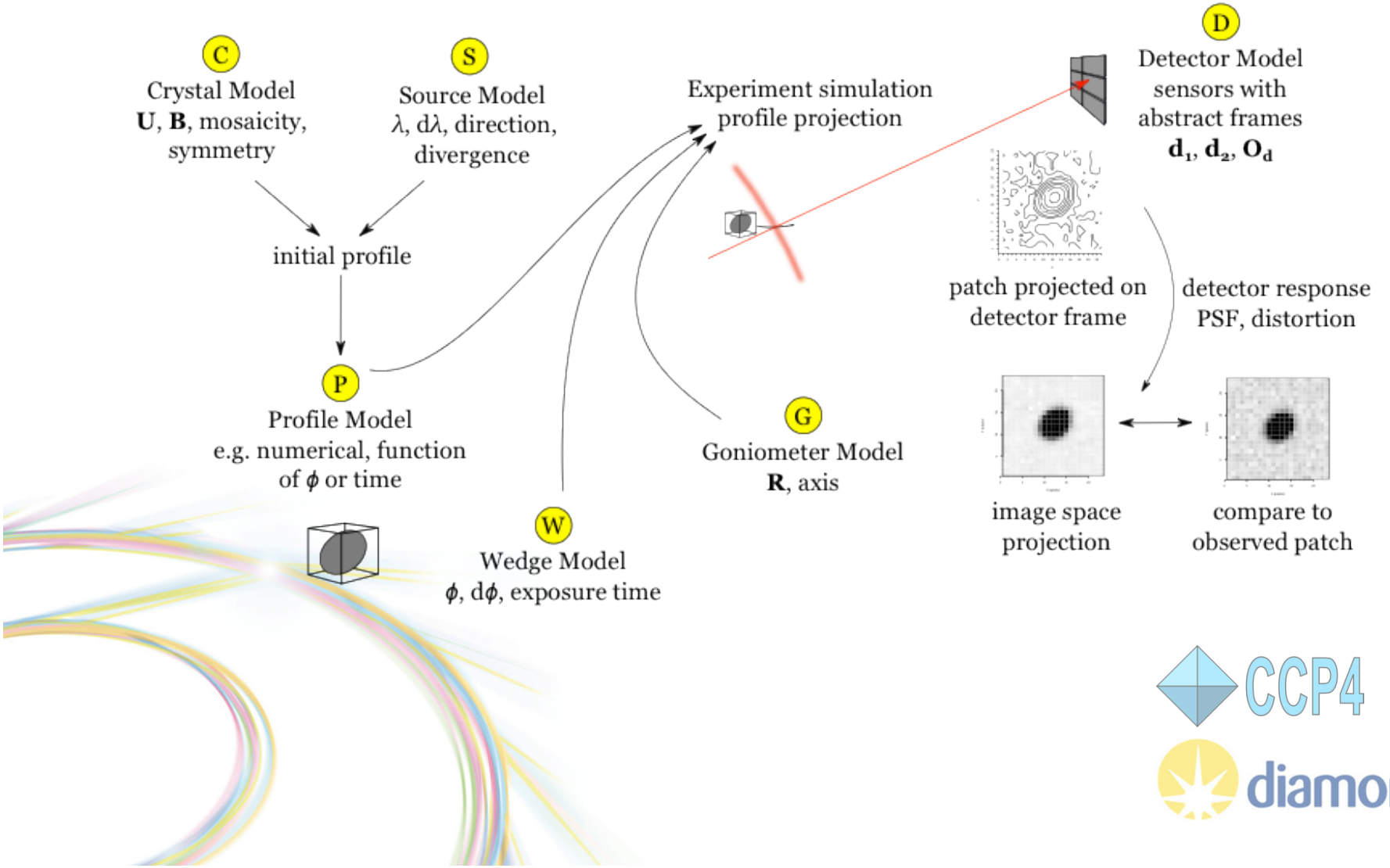
- Allow run-time customization e.g. change workflow
- Allow run-time extension with e.g. new algorithms
- Value of new algorithms can be demonstrated easily
- Development of new methods has clear starting point

Plans

- Re-implement existing methods (see James Parkhurst talk)
- Produce working version of software this year including global refinement (see David Waterman talk)
- Extend this to hybrid method in first illustration



Hybrid method



CCP4

diamond

Input from outside

- Dectris involved for proper modelling of the detector
- Anyone else can be involved to develop “plug in” detector models

CCP4

diamond

Framework Components

Fundamental data structure

ReflectionList & ObservationList

- Central data storage class: reflections indexed by miller indices, point at a list of observations which can have templated form
- Store: I, sigI, reflection profiles, partial observations, …
- Parallel operation: fork, merge: different threads operate on different instances of list

ReflectionList

- Iteration 1: map of Miller indices to:
  - Prediction
  - Shoebox
  - Ewald-sphere profile
  - I, sigI
- Iteration 2: database:
  - Peaks, profiles, predictions etc.
  - Many to many map w.r.t. Miller indices

Acknowledgements

- Support from EU FP7, Diamond Light Source, CCP4
- CCTBX for providing basis, build chain, huge amount of raw materials
- Input from many people @diamond, @LMB, @EMBL Hamburg, @CCP4 …